TABLE I
RATE $(m-r)/pm, r > 1$, QC CODES WHICH IMPROVE THE BOUNDS ON THE MAXIMUM POSSIBLE MINIMUM DISTANCE FOR A BINARY LINEAR CODE

QC Code	m	$x^m - 1/h(x)$	$d_{\min}$	d_v	$c_i(x)$
(45,10)	15	$x^5+x^4+x^2+1$	18^{d2}	17–18	2435, 175, 1557
(85,9)	17	$x^8+x^7+x^6+x^2+x+1$	39	38–40	13355, 17411, 16247, 2445, 7557
(90,13)	15	x^2+x+1	36^{d4}	35–40	7617, 2727, 17475, 25, 12163, 61
(90,14)	18	x^4+x^2+1	36^{d4}	34–39	57423, 4267, 671, 3047, 17305
(92,12)	23	$x^{11}+x^{10}+x^6+x^5+x^4+x^2+1$	38^{d2}	37–41	11436331, 5633375, 1234675, 32251
(96,13)	16	x^3+x^2+x+1	40^{d4}	38–42	7525, 15373, 2727, 6213, 273, 1271
(96,14)	16	x^2+1	38^{d2}	36–42	11511, 16637, 3633, 5113, 55, 47
(100,13)	18	x^6+x^5+x+1	40^{d4}	39–44	531235, 164131, 346777, 20441, 34741,
(105,11)	15	x^4+x^3+1	46^{d2}	45–48	17735, 2657, 25335, 2365, 7275, 1417, 7233
(108,14)	18	x^4+x^2+1	44^{d4}	42–48	67521, 275, 312563, 10327, 51443, 227
(112,9)	14	$x^5+x^4+x^3+1$	52^{d4}	51–53	1031, 1127, 273, 3535, 1675, 1313, 1763, 365
(112,10)	14	x^4+x^2+x+1	50^{d2}	48–52	553, 3423, 7471, 5333, 6563, 5577, 1441, 35

The weight distribution is

Weight	Count
0	1
18	185
20	183
22	225
24	155
26	195
28	45
30	35

These results suggest that the class of QC codes, or even just the subclass of 1-generator QC codes, will yield many more good codes.

REFERENCES

[1] T.A. Gulliver and V.K. Bhargava, "Some best rate $1/p$ and rate $(p-1)/p$ systematic quasi-cyclic codes," *IEEE Trans. Inform. Theory*, vol. IT-37, pp. 552-555, May 1991.

[2] T.A. Gulliver and V.K. Bhargava, "Nine good rate $(m-1)/pm$ quasi-cyclic codes," *IEEE Trans. Inform. Theory*, vol. IT-38, pp. 1366-1369, July 1992.

[3] G.E. Séguin and G. Drolet, "The theory of 1-generator quasi-cyclic codes," Dep. of Elec. Comput. Eng., Royal Military College of Canada, Kingston, Ont., Canada, June 1990.

[4] F.J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. New York: North-Holland, 1977.

[5] P.P. Greenough and R. Hill, "Optimal ternary quasi-cyclic codes," *Designs, Codes and Cryptography*, vol. 2, pp. 81-91, 1992.

[6] T. Verhoeff, "An updated table of minimum-distance bounds for binary linear codes," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 665-680, Sept. 1987, revised Jan. 1989.

Constructions of Skew-Tolerant and Skew-Detecting Codes

Mario Blaum, Jehoshua Bruck, and Levon H. Khachatrian

***Abstract*— Recently the paradigm of skew-tolerant parallel asynchronous communication was introduced, along with constructions for codes that can tolerate or detect skew. Some of these constructions were improved in [8]. In this paper these constructions are improved upon further, and we prove that the new constructions are, in a certain sense, optimal.**

***Index Terms*—Parallel asynchronous communication, skew, error correcting codes, unidirectional errors, t-error correcting/all unidirectional error detecting (EC/AUED) codes, pipelined transmission.**

I. INTRODUCTION

In [10], a coding solution to the problem of parallel asynchronous communications was presented. After transmission of each codeword, the receiver acknowledges reception of the message through a handshake mechanism. In this way, skew between messages is avoided. From a coding point of view, the problem is identifying the end of a message. As pointed out in [10], the codes that accomplish this task are the so called unordered codes.

A more complicated coding situation occurs when acknowledgment of the message is not allowed. In principle, this is an attractive alternative, since it would allow pipelined utilization of the channel, with increased data throughput. However, the difficulty now is that there might be skew between messages, i.e., signals from a second transmitted vector may arrive before the current vector has been completely received. Necessary and sufficient conditions for codes that can either detect or correct a certain amount of skew were given in [2]. For further motivation and description of the problem,

Mauscript received June 18, 1992; revised January 12, 1993.

M. Blaum and J. Bruck are with IBM Research Division, Almaden Research Center, San Jose, CA 95120.

L. H. Khachatrian is with the Department of Mathematics SFB 343, D-4800 Bielefeld 1 Germany. He is currrently on leave from the Institute of Problems of Informatics and Automation, Armenian Academy of Sciences.

IEEE Log number 9211278.

the reader is referred to [2], [10]. In this paper, we concentrate on constructions of codes that can detect or tolerate skew below a certain threshold.

Let us recall some of the notation given in [2]. Given two binary vectors X and Y of length n, we denote by $N(X,Y)$ the number of coordinates in which X is 1 and Y is 0. For example, if $X = 10110$ and $Y = 00101$, we have $N(X,Y) = 2$ and $N(Y,X) = 1$. Notice that $N(X,Y)+N(Y,X) = d_H(X,Y)$, where d_H denotes Hamming distance.

In [2], theorems that characterize (t_1,t_2)-skew-detecting and skew-tolerant codes were proven. Here we present the theorems in the form of definitions as follows.

Definition 1.1: Let t_1 and t_2 be two nonnegative integers, and let $t = \min\{t_1,t_2\}$ and $T = \max\{t_1,t_2\}$. We say that a binary code $\mathcal{C}$ of length n is (t_1,t_2)-skew-detecting (SD) if and only if, for any pair of distinct codewords $X,Y \in \mathcal{C}$, at least one of the following two conditions occurs: (a) $\min\{N(X,Y),N(Y,X)\} \geq t+1$ or (b) $\min\{N(X,Y),N(Y,X)\} \geq 1$ and $\max\{N(X,Y),N(Y,X)\} \geq T+1$.

Definition 1.2: Let t_1 and t_2 be two nonnegative integers, and let $t = \min\{t_1,t_2\}$. We say that a binary code $\mathcal{C}$ of length n is (t_1,t_2)-skew-tolerant (ST) if and only if, for any pair of distinct codewords $X,Y \in \mathcal{C}$, at least one of the following two conditions occurs: (a) $\min\{N(X,Y),N(Y,X)\} \geq t+1$ or (b) $\min\{N(X,Y),N(Y,X)\} \geq 1$ and $\max\{N(X,Y),N(Y,X)\} \geq t_1+t_2+1$.

As pointed out in [2], a family of (t_1,t_2)-SD or ST codes is given by the so called t-error correcting/all unidirectional error detecting (EC/AUED) codes [4]–[6], since a t-EC/AUED code satisfies condition (a) in both Definitions 1.1 and 1.2.

A generally better family of (t_1,t_2)-SD or ST codes in terms of redundancy is given by the so called error correcting unordered (ECU) codes [2], [3].

Definition 1.3: We say that a code $\mathcal{C}$ is error correcting unordered (ECU) with minimum distance d if, for any distinct $X,Y \in \mathcal{C}$, the next two conditions are satisfied:

1) $d_H(X,Y) = N(X,Y)+N(Y,X) \geq d$.
2) $\min\{N(X,Y),N(Y,X)\} \geq 1$.

The connection between ECU codes and (t_1,t_2)-SD and ST codes is given by the following lemma.

Lemma 1.1: Let t_1 and t_2 be positive integers and $t = \min\{t_1,t_2\}$. Then:

1) Let $\mathcal{C}$ be an ECU with minimum distance $\geq t_1+t_2+1$. Then $\mathcal{C}$ is (t_1,t_2)-SD.
2) Let $\mathcal{C}$ be an ECU with minimum distance $\geq t_1+t_2+t+1$. Then $\mathcal{C}$ is (t_1,t_2)-ST.

A proof of Lemma 1.1 is given in [2].

A method to construct systematic ECU codes is given next. This method is a generalization of the well known Berger construction [1]. By $\lfloor x \rfloor$ we denote the integer part of x. Also, $\lceil x \rceil = x$ if x is an integer and $\lceil x \rceil = \lfloor x \rfloor + 1$ if x is not an integer.

Construction 1.1: Assume that we want to construct an ECU code $\mathcal{C}$ with minimum distance d and dimension k. Choose an $[n',k,d]$ error correcting (EC) code $\mathcal{C}'$. Let $\underline{u}$ be an information vector of length k. Then proceed as follows.

1) Encode $\underline{u}$ into a vector $\underline{v} \in \mathcal{C}'$.
2) Let j be the Hamming weight of $\underline{v}$. Then append to $\underline{v}$ the binary representation of $\lfloor (n'-j)/d \rfloor$.

The code obtained with this encoding procedure is ECU with minimum distance d, as proven in [2]. The length of the tail added to the error-correcting code is $\lceil \log_2 \lceil (n'+1)/d \rceil \rceil$.

Improved constructions for (t_1,t_2)-ST codes were given in [8]. In the next section, we give a general method for constructing (t_1,t_2)-SD and ST codes. We show how the construction in [8] fits into the general method and we give a new construction that improves it generally by a bit. In Section III, we prove that the construction in Section II is optimal, in a sense. Section IV presents some constructions that improve upon those in Section II using ad hoc methods. We finally draw some conclusions and provide tables with parameters for some (t_1,t_2)-SD and ST codes.

II. Constructions Of Systematic (t_1,t_2)-SD And ST Codes

In this section, we construct (t_1,t_2)-SD and ST codes. Since, by Definition 1.1 (Definition 1.2), a code is (t_1,t_2)-SD (ST) if and only if it is (t_2,t_1)-SD (ST), from now on we assume, as in [8], that $t_1 \leq t_2$. The procedure involves adding three tails to the information bits: the first tail encodes the information bits into an $[n',k,2t_1+2]$ error-correcting code (for a table with the best error-correcting codes, see [11]); the second tail, to be described below, makes the code satisfy the conditions in Definitions 1.1 or 1.2; the third tail merely unorders the code. We start with a definition.

Definition 2.1: Given two positive integers w and m, a $T(w,m)$-matrix is a binary matrix with w rows $\underline{u}_0,\underline{u}_1,\ldots,\underline{u}_{w-1}$ such that, for any two rows $\underline{u}_i$ and $\underline{u}_j$, $0 \leq i < j \leq w-1$, at least one of the following two conditions occurs:

a) $N(\underline{u}_i,\underline{u}_j) \geq j-i$;

or

b) $N(\underline{u}_j,\underline{u}_i) \geq m-(j-i)$.

In what follows we present two constructions of $T(w,m)$ matrices. The first construction is implicit in the constructions of [8] while the second one, which is more efficient, is one of our contributions.

Construction 2.1: Given a vector $\underline{v}$, let $\rho(\underline{v})$ be a rotation to the right of vector $\underline{v}$. For instance, if $\underline{v} = 1100$, then $\rho(\underline{v}) = 0110$. Consider the following matrix, denoted $K(w,2l)$, with w rows $\underline{u}_0,\underline{u}_1,\ldots,\underline{u}_{w-1}$ and $2l$ columns:

1) $\underline{u}_0 = \overbrace{11\ldots1}^{l}\overbrace{00\ldots0}^{l}$.
2) For $1 \leq i \leq w-1$, $\underline{u}_i = \rho(\underline{u}_{i-1})$.

For instance, if $w = 9$ and $l = 2$, we have

$$K(9,4) = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix}.$$

It is easy to see that, for any w, the matrix $K(w,2l)$ is a $T(w,2l)$-matrix according to Definition 2.1.

Matrices with an odd number of columns are defined similarly.

Construction 2.2: Consider the $s+1$ vectors of length s $\underline{w}_0,\underline{w}_1,\ldots,\underline{w}_s$ defined as follows:

$$\underline{w}_i = \overbrace{00\ldots0}^{i}\overbrace{11\ldots1}^{s-i}.$$

Given any integer i and an integer $m > 0$, we denote by $\langle i \rangle_m$ the unique integer j, $0 \leq j \leq m-1$, such that $i \equiv j \pmod m$. Consider the following matrix, denoted $B(w,s)$, with w rows $\underline{u}_0,\underline{u}_1,\ldots,\underline{u}_{w-1}$ and s columns: row $\underline{u}_i$ is given by vector $\underline{w}_j$,

where $j = \langle i \rangle_{s+1}$. For instance, if $w = 9$ and $s = 3$, we have

$$B(9,3) = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix}.$$

We can verify that, for any w, the matrix $B(w,s)$ is a $T(w, s+1)$-matrix according to Definition 2.1.

From Constructions 2.1 and 2.2, we see that both $K(w,s)$ and $B(w, s-1)$ are $T(w,s)$-matrices. However, $B(w, s-1)$ has one column less than $K(w,s)$. This will translate into a saving of one bit in the constructions of SD and ST tolerant codes to be given below.

If $t_1 = t_2 = t$, by Definitions 1.1 and 1.3, a (t,t)-SD code is given by an ECU code with minimum distance $d = 2t+1$. However, when $t_1 < t_2$, the redundancy of a (t_1, t_2)-SD code as given by Construction 1.1 can be improved, as we show next.

Construction 2.3: Assume that we want to construct a (t_1, t_2)-SD code $\mathcal{C}$ with k information bits and $t_1 < t_2$. Choose an $[n', k, 2t_1+2]$ error correcting (EC) code $\mathcal{C}'$ in which the Hamming weights are at least two apart (e.g., all the weights are even). Consider the matrix $B(\lceil (n'+1)/2 \rceil, t_2 - t_1 - 1)$ as given by Construction 2.2. Let $\underline{u}$ be an information vector. Then proceed as follows.

1) Encode $\underline{u}$ into a vector $\underline{v} \in \mathcal{C}'$.
2) Let j be the Hamming weight of $\underline{v}$. Then append to $\underline{v}$ row $\lfloor j/2 \rfloor$ of matrix $B(\lceil (n'+1)/2 \rceil, t_2 - t_1 - 1)$.
3) Append the complement of the binary representation of $\lfloor j/(2t_1+2) \rfloor$ if $t_1 + 1 \geq t_2 - t_1$ or the complement of the binary representation of $\lfloor j/2(t_2 - t_1) \rfloor$ if $t_1 + 1 < t_2 - t_1$.

In the construction above, the matrix $B(\lceil (n'+1)/2 \rceil, t_2 - t_1 - 1)$ can be replaced by any $T(\lceil (n'+1)/2 \rceil, t_2 - t_1)$-matrix. However, we will prove in Section III that $B(w,s)$ is an optimal $T(w, s+1)$-matrix. We will prove in Theorem 1.2 that Construction 2.3 gives a (t_1, t_2)-SD code.

Example 2.1: Assume that we want to construct a $(1,5)$-SD code with 4 information bits. We start by encoding the information bits into an $[8,4,4]$ extended Hamming code. Say, we take the Hamming code with parity-check matrix

$$H = \begin{pmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

and we add a parity bit. We then append a row of matrix $B(5,3)$, where this matrix was described in Construction 2.2. Finally, we unorder the vector as described in the third step of Construction 2.3.

For instance, assume that we want to encode $\underline{u} = 0101$. The Hamming code gives $\underline{v} = 0101\ 0101$, which has weight $j = 4$. Row $j/2 = 2$ of matrix $B(5,3)$ is $\underline{u}_2 = 001$.

Now, since $t_1 + 1 = 2 < 4 = t_2 - t_1$, the third tail is given by the complement of $\lfloor j/2(t_2 - t_1) \rfloor = \lfloor 4/8 \rfloor = 0$, then the third tail is 1.

The final encoded vector is

$$\underline{c} = 0101\ 0101\ 001\ 1$$

We see that the redundancy is 8 bits. If we use Construction 1.1 together with Lemma 1.1 as in [2], we see that we need to construct an ECU code with minimum distance $d = 7$. It is easily verified that at least 12 redundant bits are needed.

Next we give an analogous construction for (t_1, t_2)-ST codes.

Construction 2.4: Assume that we want to construct a (t_1, t_2)-ST code $\mathcal{C}$ with k information bits and $t_1 \leq t_2$.

Choose an $[n', k, 2t_1+2]$ error correcting (EC) code $\mathcal{C}'$ in which the Hamming weights are at least two apart (e.g., all the weights are even). Consider the matrix $B(\lceil (n'+1)/2 \rceil, t_2 - 1)$ as given by Construction 2.2. Let $\underline{u}$ be an information vector. Then proceed as follows.

1) Encode $\underline{u}$ into a vector $\underline{v} \in \mathcal{C}'$.
2) Let j be the Hamming weight of $\underline{v}$. Then append to $\underline{v}$ row $\lfloor j/2 \rfloor$ of matrix $B(\lceil (n'+1)/2 \rceil, t_2 - 1)$.
3) Append the complement of the binary representation of $\lfloor j/(2t_1+2) \rfloor$ if $t_1 = t_2 = t$ or the complement of the binary representation of $\lfloor j/2t_2 \rfloor$ if $t_1 < t_2$.

Example 2.2: Assume that we want to construct a $(1,5)$-ST code with 4 information bits. As in Example 2.1, we start by encoding the information bits into an $[8,4,4]$ extended Hamming code. We then append a row of matrix $B(5,4)$, and then we unorder the vector as described in the third step of Construction 2.4.

As before, assume that we want to encode $\underline{u} = 0101$. The Hamming code gives $\underline{v} = 0101\ 0101$, which has weight $j = 4$. The second tail corresponds to row 2 in matrix $B(5,4)$, so it is 0011.

Since the length of the error correcting code is 8 and $\lfloor 8/10 \rfloor = 0$, the third tail is not necessary. The final encoded vector is

$$\underline{c} = 0101\ 0101\ 0011.$$

The redundancy is 8 bits. If we use Construction 1 together with Lemma 1.1 as in [2], we see that we need to construct an ECU code with minimum distance $d = 8$, and at least 13 redundant bits are needed.

We prove next that Constructions 2.3 and 2.4 provide (t_1, t_2)-SD and ST codes. We will prove a slightly more general statement.

Theorem 2.1: Let $X = (\underline{v}_1, \underline{r}_1, \underline{s}_1)$ and $Y = (\underline{v}_2, \underline{r}_2, \underline{s}_2)$ be codewords that are obtained by using Construction 2.3 with parameters a and b, $a \leq b$. Namely, $\underline{v}_1$ and $\underline{v}_2$ are even weight codewords in an $[n', k, 2a+2]$ code, $\underline{r}_1$ and $\underline{r}_2$ are the tails corresponding to rows in the $B(\lceil (n'+1)/2 \rceil, b - a - 1)$ matrix and $\underline{s}_1$ and $\underline{s}_2$ are the tails as described in the third step of the construction. Then at least one of the following two conditions occurs:

a) $\min\{N(X,Y), N(Y,X)\} \geq a + 1$

or

b) $\min\{N(X,Y), N(Y,X)\} \geq 1$ and $\max\{N(X,Y), N(Y,X)\} \geq b + 1$.

Proof: We see first that $\min\{N(X,Y), N(Y,X)\} \geq a+1$ or $\max\{N(X,Y), N(Y,X)\} \geq b+1$. Then we prove that the last tail in X and Y, namely $\underline{s}_1$ and $\underline{s}_2$, ensures that X and Y are unordered, hence, $\min\{N(X,Y), N(Y,X)\} \geq 1$. Consider $\underline{v}_1$ and $\underline{v}_2$, and, without loss of generality, let $w(\underline{v}_1) \leq w(\underline{v}_2)$, where $w(\underline{v})$ denotes the (Hamming) weight of $\underline{v}$. Moreover, let $w(\underline{v}_2) = w(\underline{v}_1) + 2l$, $l \geq 0$. If $0 \leq l \leq a$, since $d_H(\underline{v}_1, \underline{v}_2) \geq 2a + 2$, we obtain,

$$N(\underline{v}_1, \underline{v}_2) \geq a - l + 1 \tag{1}$$

$$N(\underline{v}_2, \underline{v}_1) \geq a + l + 1 \tag{2}$$

According to Construction 2.3, by the choice of $\underline{r}_1$ and $\underline{r}_2$, we have

$$N(\underline{r}_1, \underline{r}_2) \geq l \tag{3}$$

or

$$N(\underline{r}_2, \underline{r}_1) \geq b - a - l. \tag{4}$$

If inequality (3) holds, then inequalities (1) and (3) imply

$$N(X,Y) \geq N(\underline{v}_1, \underline{v}_2) + N(\underline{r}_1, \underline{r}_2) \geq a + 1, \tag{5}$$

while inequality (2) implies

$$N(Y,X) \geq N(\underline{v}_2,\underline{v}_1) \geq a+l+1 \geq a+1. \quad (6)$$

Therefore, inequalities (5) and (6) ensure that condition a) in the theorem is satisfied. If inequality (4) holds, then inequalities (2) and (4) imply

$$N(Y,X) \geq N(\underline{v}_2,\underline{v}_1) + N(\underline{r}_2,\underline{r}_1) \geq b+1.$$

Therefore, the second part of condition b) is satisfied. Now assume that $l \geq a+1$. If $2l \geq b+1$, then condition b) is satisfied, so assume that $2a+2 \leq 2l \leq b$. Therefore,

$$N(Y,X) \geq N(\underline{v}_2,\underline{v}_1) \geq 2l \geq 2a+2 > a+1. \quad (7)$$

As before, either inequality (3) or inequality (4) hold. If inequality (3) holds, then

$$N(X,Y) \geq N(\underline{r}_1,\underline{r}_2) \geq l \geq a+1, \quad (8)$$

thus, inequalities (7) and (8) imply that condition a) is satisfied. If inequality (4) holds, then

$$N(Y,X) \geq N(\underline{v}_2,\underline{v}_1) + N(\underline{r}_2,\underline{r}_1) \geq 2l + (b-a-l) = l+b-a \geq b+1,$$

so, the second part of condition b) is satisfied.

It is clear that, when the third tail is the complement of the binary representation of $\lfloor j/(2a+2) \rfloor$, j the weight of the vector $\underline{v}$ in the error-correcting code, then this tail unorders the code (this was actually proven in [2] and [3]).

So, we need to show that, when the third tail is the complement of the binary representation of $\lfloor j/2(b-a) \rfloor$, j the weight of the vector $\underline{v}$ in the error-correcting code, then this tail also unorders the code. To see this, divide the weights corresponding to the error-correcting code into sets of size $b-a$, i.e., the first set involves the weights $0, 2, \ldots, 2(b-a)-2$, the second set involves the weights $2(b-a), 2(b-a)+2, \ldots, 4(b-a)-2$, etc., the ith set involves the weights $2(i-1)(b-a), 2(i-1)(b-a)+2, \ldots, 2i(b-a)-2$. Now, observe that the second tail unorders each of these sets, since, given the ith set, to weight $2(i-1)(b-a)$ corresponds tail $\overbrace{11\ldots1}^{b-a-1}$, to weight $2(i-1)(b-a)+2$ corresponds tail $\overbrace{01\ldots1}^{b-a-1}$, etc., and to weight $2i(b-a)-2$ corresponds tail $\overbrace{00\ldots0}^{b-a-1}$. Finally, notice that the third tail is identical within a set, and it unorders the different sets. □

Based on the forgoing theorem it is easy to prove the following.

Corollary 2.1: The code obtained with Construction 2.3 is (t_1,t_2)-SD while the code obtained with Construction 2.4 is (t_1,t_2)-ST.

Proof: For (t_1,t_2)-SD codes use $a=t_1$ and $b=t_2$ in Theorem 2.1, while for (t_1,t_2)-ST codes use $a=t_1$ and $b=t_1+t_2$ in Theorem 2.1. □

The next section discusses how good the different tails are.

III. Optimality Issues

In Constructions 2.3 and 2.4, we saw that, given k information bits, we added three tails in order to obtain a (t_1,t_2)-SD or ST code. The first tail encodes the k information bits into an $[n',k,2t+2]$ EC code, where $t=\min\{t_1,t_2\}$. The second tail was a $T(\lceil (n'+1)/2 \rceil, t_2-t_1)$ or a $T(\lceil (n'+1)/2 \rceil, t_2)$ matrix, where $T(w,m)$-matrices were given by Definition 2.1. The third tail unorders the code similarly to the Berger construction [1].

In this Section, we show that the second and third tails are the shortest possible. This does not mean that Constructions 2.3 and 2.4 are globally optimal (this is an open problem), but that for these constructions, the tails are optimal. We first show that the smallest number of columns that a $T(w,s)$-matrix may have is $s-1$.

Theorem 3.1: Let T be a $T(w,m)$-matrix with s columns and $w > m$ rows. Then, $s \geq m-1$.

Proof: Let $\underline{u}_0, \underline{u}_1, \ldots, \underline{u}_{w-1}$ be the rows of matrix T. We assume that $s \leq m-2$ and reach a contradiction. Notice that

$$N(\underline{u}_i,\underline{u}_{i+1}) \geq 1 \text{ for } 0 \leq i \leq w-2. \quad (9)$$

If not, by Definition 2.1, $N(\underline{u}_{i+1},\underline{u}_i) \geq m-1$, a contradiction since $s \leq m-2$.

Claim:

$$N(\underline{u}_0,\underline{u}_i) \geq i \text{ for } 1 \leq i \leq s-2. \quad (10)$$

This is certainly true for $i=1$, by (9). Assume that there is a j, $2 \leq j \leq s-2$, that does not satisfy (10). Moreover, let j be the first with this property. Therefore,

$$N(\underline{u}_0,\underline{u}_{j-1}) \geq j-1 \quad (11)$$

and, by Definition 2.1,

$$N(\underline{u}_j,\underline{u}_0) \geq m-j. \quad (12)$$

From (11), we conclude that

$$w(\underline{u}_0) \geq j-1, \quad (13)$$

where $w(\underline{u})$ denotes the Hamming weight of vector $\underline{u}$. From (12), we conclude that the number of zeros in $\underline{u}_0$, i.e., $s-w(\underline{u}_0)$, is at least $m-j$. Therefore, $w(\underline{u}_0) \leq s-m+j$, and since $s \leq m-2$, we obtain

$$w(\underline{u}_0) \leq j-2. \quad (14)$$

But (13) and (14) contradict each other, so (10) is true. In particular, $N(\underline{u}_0,\underline{u}_{s-2}) \geq s-2$, and since each row $\underline{u}_i$ has length at most $s-2$, $\underline{u}_{s-2}$ is the all-zero vector. On the other hand, by (9), $N(\underline{u}_{s-2},\underline{u}_{s-1}) \geq 1$, a contradiction. □

Theorem 3.1 shows that matrix $B(w,s-1)$ is an optimal $T(w,s)$-matrix.

We conclude this section by describing a method for optimal unordering of error-correcting codes. These results generalize the ones given in [1] and in [7]. Let us start with a definition.

Definition 3.1: Let $V=\{\underline{v}_1,\underline{v}_2,\ldots,\underline{v}_m\}$ be a set of m binary vectors. We say that V is a chain of length m if $\underline{v}_1 \subset \underline{v}_2 \subset \ldots \subset \underline{v}_m$, i.e., $N(\underline{v}_i,\underline{v}_{i+1})=0$ for $1 \leq i \leq m-1$.

Lemma 3.1 Let $\mathcal{C}$ be a binary code of length n, and let m be the maximal length of a chain in $\mathcal{C}$. Then

1) The tail we need to append to codewords in $\mathcal{C}$ to make $\mathcal{C}$ unordered is of length at least $\lceil \log_2 m \rceil$ bits.
2) Code $\mathcal{C}$ can be made unordered by appending to each codeword a tail of length $\lceil \log_2 m \rceil$ bits.

TABLE I
PARAMETERS OF SOME $(1,2)$-ST (OR $(1,3)$-SD) CODES

k	$n-k$ from [2]	$n-k$ from [8]	$n-k$ from Constr. 2.4
8	10	9	8
16	13	11	10
24	14	11	10
32	15	13	12
64	17	15	14

TABLE II
PARAMETERS OF SOME $(1,3)$-ST (OR $(1,4)$-SD) CODES

k	$n-k$ from [2]	$n-k$ from [8]	$n-k$ from Constr. 2.4	$n-k$ from Section IV
8	11	10	9	8
16	14	12	10	
24	15	12	11	
32	15	14	12	
64	18	16	14	

TABLE III
PARAMETERS OF SOME $(1,4)$-ST (OR $(1,5)$-SD) CODES

k	$n-k$ from [2]	$n-k$ from [8]	$n-k$ from Constr. 2.4	$n-k$ from Section IV
8	13	11	9	
16	17	13	11	10
24	19	13	11	
32	20	15	13	12
64	23	17	15	14

TABLE IV
PARAMETERS OF SOME $(2,2)$-ST (OR $(2,4)$-SD) CODES

k	$n-k$ from [2]	$n-k$ from [8]	$n-k$ from Constr. 2.4	$n-k$ from Section IV
8	13	13	12	11
16	17	15	15	14
24	19	17	16	
32	20	18	17	
64	23	21	20	

Proof: We first prove the lower bound. Let $\underline{v}_1, \underline{v}_2, \ldots, \underline{v}_m$ be a chain of maximal length m. Let $\underline{t}_1, \underline{t}_2, \ldots, \underline{t}_m$ be a set of tails such that the vectors $(\underline{v}_1, \underline{t}_1), (\underline{v}_2, \underline{t}_2), \ldots, (\underline{v}_m, \underline{t}_m)$ are unordered. Clearly, the $\underline{t}_i$'s must all be distinct, if not, the vectors $(\underline{v}_i, \underline{t}_i)$ would not be unordered. Therefore, the number of bits we have to add to unorder $\mathcal{C}$ is at least $r = \lceil \log_2 m \rceil$ bits. We show next that this number is also sufficient by giving a constructive procedure for unordering the code.

Let $w_1 < w_2 < \ldots < w_l$ be the weight spectrum (the set of possible weights) of code $\mathcal{C}$ and let $\mathcal{C}_{w_i}$ be the set of codewords of weight w_i, $1 \le i \le l$. Denote by σ_j the binary representation of j, $0 \le j \le m-1$, as a vector of length r. Next, we assign a tail σ_i to each $\underline{c} \in \mathcal{C}$ such that the resulting vectors $(\underline{c}, \sigma_i)$ are unordered. The assignment of tails is done as follows:

1) Assign σ_0 to each codeword in $\mathcal{C}_{w_l}$ (notice that $\sigma_0 = \overbrace{00\ldots0}^{r}$).
2) Assume that sets $\mathcal{C}_{w_l}, \mathcal{C}_{w_{l-1}}, \ldots, \mathcal{C}_{w_j}$, $j \ge 1$, have been assigned tails, and the last assigned tail is σ_i, $i \le m-1$. If either $j = 1$ (all sets have been assigned a tail) or $i = m-1$ (exactly m tails have been assigned), then stop. Otherwise, assign tail σ_i to the codewords in $\mathcal{C}_{w_{j-1}}$ that are not contained in any codeword that has been assigned tail σ_i, and tail σ_{i+1} to the rest of the codewords in $\mathcal{C}_{w_{j-1}}$, if any.

The set of vectors $(\underline{c}, \sigma_i)$ is unordered. In effect, let $(\underline{c}, \sigma_i)$ and $(\underline{c}', \sigma_j)$ be any two of these vectors, and assume that $w(\underline{c}) \ge w(\underline{c}')$. Therefore, by construction, $i \le j$. If $i < j$, then $N(\sigma_j, \sigma_i) > 0$, and since $N(\underline{c}, \underline{c}') > 0$, then $(\underline{c}, \sigma_i)$ and $(\underline{c}', \sigma_j)$ are unordered. If $i = j$, by construction, $\underline{c}$ and $\underline{c}'$ are unordered.

We complete the proof by showing that the inductive process described above assigns a tail to each codeword in $\mathcal{C}$. Therefore, the number of assigned tails is at most m. In effect, assume that some codewords have not been assigned tails, and that $\mathcal{C}_{w_s}$ is the first set having codewords that have not been assigned a tail. In particular, some codewords have been assigned tail σ_{m-1} (the last tail). We will reach a contradiction by constructing a chain of length $m+1$ in $\mathcal{C}$. We proceed by induction as follows:

1) Choose as the first element of the chain a codeword $\underline{c}_0 \in \mathcal{C}_{w_s}$ that has not been assigned a tail.
2) Assume that we have obtained the chain $\underline{c}_0 \subset \underline{c}_1 \subset \ldots \subset \underline{c}_j$, such that $\underline{c}_1$ has been assigned σ_{m-1}, $\underline{c}_2$ has been assigned σ_{m-2}, etc., and $\underline{c}_j$ has been assigned σ_{m-j}. If $j = m$ then stop. Otherwise, by construction, $\underline{c}_j$ is contained in some $\underline{c} \in \mathcal{C}$ that has been assigned σ_{m-j-1}. Denote this $\underline{c}$ by $\underline{c}_{j+1}$ and add it to the chain.

We have obtained a chain $\underline{c}_0 \subset \underline{c}_1 \subset \ldots \subset \underline{c}_m$ in $\mathcal{C}$ with $m+1$ elements, contradicting the maximality of m. Therefore, each codeword in $\mathcal{C}$ has been assigned a tail. □

We note here that the proof of Lemma 3.1 provides an algorithm for unordering the code as well as an algorithm for finding a maximal chain. In [4], it was proven that the $[2^m, 2^m - m - 1, 4]$ extended Hamming code contains a chain of $2^{m-2}+1$ codewords. Therefore, in order to make this code unordered, we need to append a tail of length $\lceil \log_2(2^{m-2}+1) \rceil = m-1$. However, we can easily see that a coset of this code contains chains of length at most 2^{m-2}. Hence, using Lemma 3.1, we have the following.

Corollary 3.1: Let $\mathcal{C}'$ be any coset of a $[2^m, 2^m - m - 1, 4]$ extended Hamming code. Then, it is sufficient to add a tail of length $m-2$ to $\mathcal{C}'$ to make it unordered.

IV. FURTHER IMPROVEMENTS

In this section, we present some further improvements to the results of Section II. The idea is to unify the second and the third tails, and exploit the weight distribution of the particular code considered with ad hoc methods. Some of the codes obtained with the methods of Sections II and IV are tabulated in Tables I to VI.

Consider for instance a (1,3)-ST with $k = 8$ information bits, as in the first row of Table II. In order to encode into a code of minimum distance 4, we need to add 5 extra redundant bits, giving the following weight distribution: 0, 4, 6, 8, 10, 12. For the second tail, we can use the following assignment:

$$0 \leftrightarrow 10$$
$$4 \leftrightarrow 01$$
$$6 \leftrightarrow 00$$
$$8 \leftrightarrow 11$$
$$10 \leftrightarrow 01$$
$$12 \leftrightarrow 00$$

TABLE V
PARAMETERS OF SOME $(2,3)$-ST (OR $(2,5)$-SD) CODES

k	$n-k$ from [2]	$n-k$ from [8]	$n-k$ from Constr. 2.4	$n-k$ from Section IV
8	14	14	13	12
16	17	17	16	
24	20	18	17	16
32	21	19	18	
64	24	22	21	

TABLE VI
PARAMETERS OF SOME $(3,3)$-ST (OR $(3,6)$-SD) CODES

k	$n-k$ from [2]	$n-k$ from [8]	$n-k$ from Constr. 2.4	$n-k$ from Section IV
8	20	16	16	15
16	22	20	19	18
24	25	22	22	21
32	28	24	23	22
64	32	27	27	26

For the third tail (unordering the code), we can simply assign 1 to weights 0, 4 and 6, and 0 to weights 8, 10 and 12. Hence, the total redundancy is 8 bits, as opposed to 9 bits using Construction 2.4.

Consider a (1,4)-ST code with $k = 20$ information bits. We need to add 6 extra redundant bits in order to obtain a shortened extended Hamming code with a codeword of weight 26. The weight distribution is $0, 4, 6, \ldots, 20, 22, 26$. We can verify that the following assignment for the tail gives a (1,4)-ST code:

$$
\begin{aligned}
0 &\leftrightarrow 1111 \\
4 &\leftrightarrow 0011 \\
6 &\leftrightarrow 1110 \\
8 &\leftrightarrow 1101 \\
10 &\leftrightarrow 0101 \\
12 &\leftrightarrow 0110 \\
14 &\leftrightarrow 1010 \\
16 &\leftrightarrow 1001 \\
18 &\leftrightarrow 0001 \\
20 &\leftrightarrow 0010 \\
22 &\leftrightarrow 1100 \\
26 &\leftrightarrow 0000
\end{aligned}
$$

The total redundancy is 10 bits, as opposed to 11 bits using Construction 2.4. By using part of the assignment above for $k = 16$, we can also lower $n - k$ from 11 to 10, as shown in Table III.

If we take a (1,4)-ST with $k = 47$ information bits, we need to add 7 redundant bits in order to obtain a shortened extended Hamming code with a codeword of weight 54. The weight distribution is $0, 4, 6, \ldots, 48, 50, 54$. We can verify that the following assignment for the tail gives a (1,4)-ST code:

$$
\begin{aligned}
0 &\leftrightarrow 11111 \\
4 &\leftrightarrow 11100 \\
6 &\leftrightarrow 11011 \\
8 &\leftrightarrow 10111 \\
10 &\leftrightarrow 00111 \\
12 &\leftrightarrow 01011 \\
14 &\leftrightarrow 11010 \\
16 &\leftrightarrow 10101 \\
18 &\leftrightarrow 01101 \\
20 &\leftrightarrow 01110 \\
22 &\leftrightarrow 10110 \\
24 &\leftrightarrow 10011 \\
26 &\leftrightarrow 11001 \\
28 &\leftrightarrow 11000 \\
30 &\leftrightarrow 10100 \\
32 &\leftrightarrow 00101 \\
34 &\leftrightarrow 01010 \\
36 &\leftrightarrow 10010 \\
38 &\leftrightarrow 10001 \\
40 &\leftrightarrow 01001 \\
42 &\leftrightarrow 01100 \\
44 &\leftrightarrow 00110 \\
46 &\leftrightarrow 10000 \\
48 &\leftrightarrow 01000 \\
50 &\leftrightarrow 00011 \\
54 &\leftrightarrow 00000
\end{aligned}
$$

The total redundancy is 12 bits, as opposed to 13 bits using Construction 2.4. By using part of the assignment above for $k = 32$, we can also lower $n - k$ from 13 to 12, as shown in Table III.

If we observe the constructions above, we can see that in general, for a tail of length m, $m \geq 4$, we construct a matrix with rows $\underline{r}_0, \underline{r}_1, \underline{r}_2, \ldots$ such that: $\underline{r}_0$ is the all-1 vector (corresponding to a codeword of weight 0 in the error-correcting code with $d = 4$); $\underline{r}_1$ has weight $m - 2$ (corresponding to weight 4); $\underline{r}_2$ and $\underline{r}_3$ have weight $m - 1$ (corresponding to weights 6 and 8, respectively) and $N(\underline{r}_3, \underline{r}_1) = 2$; we then add all possible vectors of weight $m-2, m-3, \ldots, 3$, such that either $N(\underline{r}_j, \underline{r}_{j+2}) \geq 2$ or $N(\underline{r}_{j+2}, \underline{r}_j) \geq 2$; we then add all the vectors of weight 2 (except one) with the same properties; finally, we add two vectors of weight 1, the remaining vector of weight 2, and the all-0 vector. As we can see, with this procedure we obtain $2^m - 2(m-2)$ vectors. For instance, if $m = 6$, we obtain a matrix with 56 vectors $\underline{r}_0, \underline{r}_1, \ldots, \underline{r}_{55}$. If we encode the data into a $[114, 106, 4]$ code, the weight distribution is $0, 4, 6, \ldots, 108, 110, 114$. Therefore, $\underline{r}_0$ is appended to the codeword of weight 0, $\underline{r}_1$ to codewords of weight 4, $\underline{r}_2$ to codewords of weight 6, etc., and $\underline{r}_{53}$ is appended to codewords of weight 108, $\underline{r}_{54}$ to codewords of weight 110, and $\underline{r}_{55}$ to the codeword of weight 114. Hence, this gives a (1,4) ST code with $k = 106$ and $n - k = 14$. In particular, we can use the vectors $\underline{r}_i$ for smaller values of k, like

$k = 64$, improving the value of $n-k$ obtained using Construction 2.4.

Consider (2,2)-ST codes. If $k = 8$, we encode the 8 information bits into a $[17, 8, 6]$ code [11]. The weight distribution of this code is 0,6,8,10,16. Therefore, we can make the following assignment:

$$0 \leftrightarrow 11$$
$$6 \leftrightarrow 01$$
$$8 \leftrightarrow 10$$
$$10 \leftrightarrow 01$$
$$16 \leftrightarrow 00$$

Thus, we improve the value in the first row of Table IV from 12 to 11. Similarly, for $k = 16$, we encode the 16 information bits into a $[25, 16, 6]$ code [11], and there is a weight assignment that improves the second row of Table IV from 15 to 14.

For (2,3)-ST codes and $k = 8$, we make the following assignment:

$$0 \leftrightarrow 111$$
$$6 \leftrightarrow 011$$
$$8 \leftrightarrow 101$$
$$10 \leftrightarrow 110$$
$$16 \leftrightarrow 000$$

This improves the value of $n - k$ in the first row of Table V from 13 to 12. Similarly, for $k = 16$, there is a weight assignment that improves $n - k$ in the third row of Table V from 17 to 16.

Finally, consider (3,3)-ST codes. For $k = 8$, there is a $[20, 8, 8]$ code (shortened extended Golay) with weight distribution 0,8,10,12,16. To this code, we can make the assignment

$$0 \leftrightarrow 111$$
$$8 \leftrightarrow 011$$
$$10 \leftrightarrow 101$$
$$12 \leftrightarrow 110$$
$$16 \leftrightarrow 000$$

which improves the value of $n - k$ in the first row of Table 6 from 16 to 15. For $k = 16$, there is a $[31, 16, 8]$ code. The following assignment gives a (3,3)-ST code.

$$0 \leftrightarrow 111$$
$$8 \leftrightarrow 011$$
$$10 \leftrightarrow 101$$
$$12 \leftrightarrow 110$$
$$14 \leftrightarrow 011$$
$$16 \leftrightarrow 101$$
$$18 \leftrightarrow 110$$
$$20 \leftrightarrow 010$$
$$22 \leftrightarrow 001$$
$$24 \leftrightarrow 100$$
$$26 \leftrightarrow 010$$
$$28 \leftrightarrow 001$$
$$30 \leftrightarrow 100$$

which improves the value of $n - k$ in the second row of Table 6 from 19 to 18. Similarly, for $k = 24$, we can improve $n - k$ from 22 to 21, for $k = 32$ we can improve $n - k$ from 23 to 22, and for $k = 64$ we can improve $n - k$ from 27 to 26.

V. Concluding Remarks

In this section, we draw some conclusions and present tables of (t_1, t_2)-SD and ST codes.

We make the following observations.

1) Given an $[n', k, 2t_1 + 2]$ code, Construction 2.3 requires an additional $t_2 - t_1 - 1 + \lceil \log_2 \lceil \min\{(n'+1)/(2t_1+2);\ (n'+1)/2(t_2 - t_1)\} \rceil \rceil$ bits to obtain a (t_1, t_2)-SD code.
2) Given an $[n', k, 2t_1 + 2]$ code, Construction 2.4 requires an additional $t_2 - 1 + \lceil \log_2 \lceil \min\{(n' + 1)/(2t_1 + 2);\ (n' + 1)/(2t_2)\} \rceil \rceil$ bits to obtain a (t_1, t_2)-ST code.
3) Constructions 2.3 and 2.4 can be made more efficient by taking a coset of an $[n', k, 2t_1+2]$ code instead of the code itself. This way, the spread of the weight distribution is reduced and the tables given below can be slightly improved. For more details about this technique, see [6].
4) The redundancy of (t_1, t_2)-ST codes given in [8] in addition to the $(n', k, 2t_1 + 2)$ code is $t_2 + \lceil \log_2 \lceil (n'+1)/(2t_1+2) \rceil \rceil$ if $t_1 < t_2$ and $t + \lceil \log_2 \lceil (n' + 1)/(4t) \rceil \rceil$ if $t_1 = t_2 = t$. Therefore, Construction 2.4 improves it by a bit when $t_1 < t_2$, and it either ties it or improves it by a bit when $t_1 = t_2$.
5) It is easy to prove from Definitions 1.1 and 1.2 that (without loss of generality, $t_2 \geq t_1$) a (t_1, t_2)-ST code is also a $(t_1, t_1 + t_2)$-SD code. Conversely, a (t_1, t_2)-SD code with $2t_1 \leq t_2$ is a $(t_1, t_2 - t_1)$-ST code. This observation is exemplified in Tables I to VI.

Tables I to VI give the redundancy (denoted $n - k$) of some (t_1, t_2)-SD and ST codes for different values of the number of information bits k using Constructions 2.3 and 2.4. The parameters of the error-correcting codes used are taken from [11]. We also include in the tables the values of the redundancy obtained with the constructions given in [2] and [8].

Acknowledgment

We are indebted to S. Wuytack for observations that allowed to improve the third tail in Constructions 2.3 and 2.4. Many useful suggestions were provided by the referees.

References

[1] J. M. Berger, "A note on error detecting codes for asymmetric channels," *Inform. and Contr.*, vol. 4, pp. 68–73, Mar. 1961.
[2] M. Blaum and J. Bruck, "Coding for skew-tolerant parallel asynchronous communications," also in *IEEE Trans. Inform. Theory*, vol. 39, Mar. 1993.
[3] M. Blaum and J. Bruck, "Unordered error-correcting codes and their applications," FTCS-92, Digest of Papers, pp. 486–493, July 1992.
[4] M. Blaum and H. van Tilborg, "On t-error correcting/all unidirectional error detecting codes," *IEEE Trans. Comput.*, vol. 38, pp. 1493–1501, Nov. 1989.
[5] F. J. H. Boinck and H. van Tilborg, "Constructions and bounds for systematic tEC/AUED codes," *IEEE Trans. Inform. Theory*, vol. 36, pp. 1381–1390, Nov. 1990.
[6] J. Bruck and M. Blaum, "New techniques for constructing EC/AUED codes," *IEEE Trans. Comput.*, vol. C-41, pp. 1318–1324, Oct. 1992.
[7] C. V. Freiman, "Optimal error detecting codes for completely asymmetric binary channels," information and control, vol. 5, pp. 64–71, Mar. 1962.
[8] L. H. Khachatrian, "Construction of (t_1, t_2)-tolerant Codes," in *Proc. Dilijan Conf.*, Sept. 1991.
[9] F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*. Amsterdam, The Netherlands: North-Holland, 1977.
[10] T. Verhoeff, "Delay-insensitive codes —An overview," *Distrib. Comput.*, vol. 3, pp. 1–8, 1988.
[11] T. Verhoeff, "An updated table of minimum distance bounds for binary linear codes," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 665–680, Sept. 1987.